WHAT IS CLAIMED IS:

1. A processor that predicts aliasing between read type instructions and write type instructions based at least in part on respective displacements between the read type and write type instructions and on previous detection of respective aliasings between the read type instructions and the write type instructions, and that bypasses data from the write type instructions to the corresponding predicted to alias read type instructions using register information of the write type instructions.

2. The processor of claim 1 wherein the data is bypassed if a threshold number of repeated aliases are detected between read type instructions and write type instructions with the respective displacements.

3. The processor of claim 1 that includes encodings of read type instruction information, write type instruction information, and repeat aliasing.

4. The processor of claim 3 wherein the encodings comprise:

a read type instruction aliasing predictor table that indicates a static instruction identifier for a read type instruction, a displacement between the indicated read type instruction and a previously aliased write type instruction, and an alias prediction confidence indicator that indicates confidence of alias predictions;

a write type instruction aliasing predictor table that indicates the static instruction identifier for a write type instruction; and

an aliasing predictor management table that indicates a rename register identifier, an alias prediction counter that indicates a number of alias predictions, and a dynamic instruction identifier.

5. The processor of claim 4 wherein the static instruction identifier includes an address for the read or write type instruction.

6. The processor of claim 4 wherein the dynamic instruction identifier monotonically increases with execution of a program that includes the instructions.

7. The processor of claim 4 that reduces the alias confidence prediction indicator for a read type instruction indicated in the read type instruction aliasing predictor table if the aliasing predictor management table does not indicate a write type instruction that corresponds to the read type instruction and the read type instruction's corresponding displacement.

8. The processor of claim 4 that reduces the alias confidence prediction indicator for a read type instruction indicated in the read type instruction aliasing predictor table if a misprediction of the read type instruction occurs.

9. The processor of claim 4 wherein the read type instruction aliasing predictor table includes a validity flag that indicates whether a threshold number of aliasings have been detected.

10. The processor of claim 1 wherein data bypasses comprise the processor substituting a register move instruction for the read type instruction.

11. The processor of claim 10, wherein a loadCheck instruction is inserted, which when executed by the processor, causes the processor to verify the predicted aliasing.

12. The processor of claim 10 wherein the register move instruction includes an integer-to-integer move instruction, a floating point-to-floating point move instruction, an integer-to-floating point move instruction, and a floating point-to-integer move instruction.

13. The processor of claim 1 wherein data bypasses comprise the processor mapping the read type instruction's destination register to the write type instruction's source register.

14. The processor of claim 13 that replaces the read type instruction with a loadCheck instruction, which when executed by the processor, causes the processor to verify the predicted aliasing.

15. The processor of claim 14 wherein the processor's verification of the predicted aliasing comprises interrogation of a data hazard detection module to ascertain whether addresses of the write type instruction and the read type instruction predicted to alias with the write type instruction match, and verification of absence of intervening matching write type instructions.

16. A method comprising:

in a register rename stage, tracking a write type instruction that has previously been indicated as aliased; and

predicting a read type instruction will alias with the write type instruction if displacement between the read type instruction and the write type instruction matches displacement between the read type instruction and a previously aliased write type instruction.

17. The method of claim 16 wherein the displacement is measured with dynamic instruction identifiers, wherein the dynamic instruction identifiers identify instances of instructions with respect to program execution.

18. The method of claim 16 wherein the write type instruction and the read type instruction are tracked with their static identifier, wherein the static identifier identifies an instruction in a program and remains static during program execution.

19. The method of claim 18 wherein the static identifier includes an instruction address.

20. The method of claim 16 wherein a read type instruction includes a load instruction, a load halfword instruction, a load byte instruction , a load float instruction , a load double instruction, and a load multiple instruction.

21. The method of claim 16 wherein the write type instruction includes a store instruction, a store byte instruction, a store float instruction, a store double instruction, a store multiple instruction, and a store halfword instruction.

22. The method of claim 16 further comprising bypassing data of the write type instruction to the read type instruction with register information of the write type instruction.

23. The method of claim 22 wherein bypassing comprises mapping the read type instruction's destination register to the write type instruction's source register.

24. The method of claim 23 further comprising replacing the read type instruction with a loadCheck instruction, wherein the loadCheck instruction causes interrogation of a data hazard detection module to ascertain whether addresses of the write type instruction and the read type instruction predicted to alias with the write type instruction match, and to ascertain whether there are any intervening matching write type instructions.

25. The method of claim 22 wherein bypassing comprises converting the read type instruction to a register move instruction.

26. The method of claim 25 further comprising inserting a loadCheck instruction, wherein the loadCheck instruction causes interrogation of a data hazard detection logic to ascertain whether addresses of the write type instruction and the read type instruction predicted to alias with the write type instruction match, and to ascertain whether there are any intervening matching write type instructions.

27. The method of claim 16 embodied as a computer program product encoded in one or more machine-readable media.

28. A method comprising:

observing repeated aliasing between a first write type instruction and a read type instruction based at least in part on their static identifiers;

determining a displacement between the first write type instruction and the read type instruction based on dynamic identifiers of the instructions;

predicting aliasing between the read type instruction as identified by the static identifier thereof and a second write type instruction identified with a

dynamic identifier that corresponds to the read type instruction's dynamic identifier and the displacement.

29. The method of claim 28 further comprising bypassing data of the second write type instruction to the read type instruction with register information of the second write type instruction.

30. The method of claim 29 wherein bypassing the data comprises mapping the data source of the read type instruction to the data source of the write type instruction.

31. The method of claim 30 further comprising substituting a loadCheck instruction for the read type instruction, wherein execution of the loadCheck instruction causes interrogation of a data hazard detection module to ascertain whether addresses of the read type instruction and the second write type instruction match.

32. The method of claim 31 wherein execution of the loadCheck instruction further causes verifying the absence of intervening matching write type instructions.

33. The method of claim 29 wherein bypassing the data comprises substituting a register move instruction for the read type instruction, wherein the move instruction moves data from the data source of the write type instruction to the data destination of the read type instruction.

34. The method of claim 28 wherein the dynamic identifiers monotonically increase with execution of a program that includes the instructions.

35. The method of claim 28 wherein the static identifiers include instruction addresses.

36. The method of claim 28 wherein the aliasing is predicted if the number of observed repeat aliasings exceeds a threshold.

37. The method of claim 28 embodied as a computer program product encoded in one or more machine-readable media.

38. A method comprising:

detecting aliasing between a read type instruction and a first write type instruction;

determining displacement between the read type instruction and the first write type instruction, wherein the displacement is with respect to program execution;

observing repeated aliasing between the read type instruction and the first write type instruction;

selecting a second write type instruction based at least in part on the displacement and the read type instruction; and

bypassing data from the second write type instruction's data source to the read type instruction's data destination.

39. The method of claim 38 further comprising verifying that the second write type instruction aliases with the first read type instruction.

40. The method of claim 38 wherein data bypass comprises changing the read type instruction to a register move instruction.

41. The method of claim 40 further comprising inserting a loadCheck instruction, wherein the loadCheck instruction causes verification that the read type instruction and the second write type instruction alias and verification of the absence of one or more intervening write type instructions.

42. The method of claim 38 wherein data bypasses comprise mapping the read type instruction's architectural destination register to the second write type instruction's rename source register.

43. The method of claim 42 further comprising changing the read type instruction to a loadCheck instruction, wherein the loadCheck instruction causes

verification that the read type instruction and the second write type instruction alias and verification of the absence of one or more intervening write type instructions.

44. The method of claim 38 wherein the first and second write type instructions have a same static identifier and different dynamic identifiers.

45. The method of claim 44 wherein the static identifiers include instruction addresses.

46. The method of claim 38 wherein the displacement is based at least in part on the dynamic identifiers of the instructions, wherein the dynamic identifiers monotonically increase with execution of a program that includes the instructions.

47. The method of claim 38 embodied as a computer program product encoded in one or more machine-readable media.

48. A computer program product encoded in one or more machine-readable media, the computer program product comprising:

a first sequence of instructions to,

update a first encoding with a read type instruction's static identifier and a displacement between the read type instruction and a write type instruction observed as aliasing with the read type instruction if the read type instruction's static identifier is not indicated in the first encoding and to update the first encoding to indicate repeat aliasing if the read type instruction's static identifier is already indicated in the first encoding,

update a second encoding with the write type instruction's static identifier;

a second sequence of instructions to update a third encoding with a write type instruction's dynamic identifier if the static identifier thereof is indicated in the second encoding; and

a third sequence of instructions to bypass data from a write type instruction to a read type instruction with register information of the write type

instruction based at least in part on displacement between the dynamic identifiers of the write type instruction and the read type instruction.

49. The computer program product of claim 48 wherein data bypassing comprises the third sequence of instructions to map the read type instruction's destination register to the write type instruction's source register.

50. The computer program product of claim 49 wherein the read type instruction's destination register includes an architectural register and the write type instruction's source register includes a rename register.

51. The computer program product of claim 50 wherein the read type instruction is replaced with a loadCheck instruction, which when executed causes verification that the read type instruction and the write type instruction alias to the same address and verification of the absence of one or more intervening write type instructions aliasing to the same address.

52. The computer program product of claim 48 wherein bypassing data comprises replacing the read type instruction with a register move instruction.

53. The computer program product of claim 52, further comprising inserting a loadCheck instruction proximate with the register move instruction, wherein execution of the loadCheck instruction causes verification that the read type instruction and the write type instruction alias to the same address and verification of the absence of one or more intervening write type instructions aliasing to the same address.

54. An apparatus comprising:

a data hazard detection module; and

means for predicting aliasing between a read type instruction and a first write type instruction based on displacement between the instructions and previously observed aliasing between the read type instruction and a second write type instruction, wherein the read type instruction and the second write type instruction also have the same displacement.

55. The apparatus of claim 54 further comprising means for bypassing data from the write type instruction to the read type instruction with register information of the write type instruction.

56. The apparatus of claim 54 wherein the read type instruction includes a load instruction, a load halfword instruction, a load byte instruction, a load float instruction, a load double instruction, and a load multiple instruction.

57. The apparatus of claim 54 wherein the write type instruction includes a store instruction, a store byte instruction, a store float instruction, a store double instruction, a store multiple instruction, and a store halfword instruction.

58. An apparatus comprising:

a data hazard detection module; and

an instruction rename unit coupled with the data hazard detection module, the instruction rename unit to rename registers of instructions and to predict aliasing between read type instructions and write type instructions based at least in part on respective displacements between the instructions, wherein the instruction rename unit includes one or more structures to,

track read type instructions indicated by the data hazard detection module as aliasing and repeat aliasing of the tracked read type instructions, and to indicate displacements between the tracked read type instructions and aliased write type instructions;

indicate write type instructions indicated by the data hazard detection module as aliasing; and

indicate write type instructions encountered in the rename unit that are indicated in the second structure.

59. The apparatus of claim 58 wherein the data hazard detection module includes a memory disambiguation buffer or a memory disambiguation buffer.

60. The apparatus of claim 58 further comprising an instruction scheduling unit coupled with the instruction rename unit and the data hazard detection module.

61. The apparatus of claim 58 wherein the structures include hardware tables and logical structures instantiable in memory.

62. An apparatus comprising:

an alias predictor,

including one or more encodings to host indications of particular instances of write type instructions and particular instances of read type instructions, respective execution displacements between respective ones of the particular instances of read and write type instructions, and register information of the particular write type instruction instances,

to predict aliasings between read and write type instruction instances based, at least in part, on the encodings and indications of detected aliasings between the instruction instances;

a rename unit coupled with the alias predictor, the rename unit to supply register information for write type instruction instances to the alias predictor; and

a data hazard detection unit coupled with the alias predictor, the data hazard detection unit to detect aliasing between particular instances of read and write type instructions and to indication said detections to the alias predictor.

63. The apparatus of claim 62, wherein the indications of particular instances of instructions include instruction instance addresses.

64. The apparatus of claim 63, wherein the instruction instances addresses include one or more of static identifiers and dynamic identifiers.

65. The apparatus of claim 62, wherein the encodings comprise:

a first encoding to indicate particular instances of read type instructions with static identifiers thereof, respective execution displacements with potentially aliasing instances of write type instructions, and respective alias prediction confidence;

a second encoding to indicate particular instances of write type instructions with static identifiers thereof; and

a third encoding to indicate particular instances of write type instructions with dynamic identifiers and register information thereof.

66. The apparatus of claim 65 further comprising:

the first and second encodings to also indicate alias prediction validity; and

the third encoding to also indicate pending unverified alias predictions.